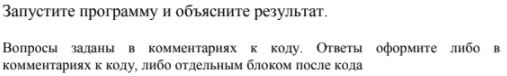
Лабораторная работа №6. Битовые (поразрядные) операции.

Задание 1.



Код:

#include<stdio.h>

#include<conio.h>

#include<windows.h>

**void** main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

**int** i,a,b; *// Сколько (битов/байтов) памяти занимают a и b?*

*/\* 4 байта в соответствии с моделью данных LLP64, используемой в  
Windows \*/*

printf("**\n**Введите два целых числа: "); scanf("%d %d",&a,&b);

printf("**\n**");

*/\* ———————————————————————————————————— \*/*

printf("Операция 'инверсия битов' в числе a: 0x%x=%d**\n**", ~a,~a); */\* Объясните*

*запись 0x%x=%d \*/*

printf("Операция 'инверсия битов' в числе b: 0x%x=%d**\n**", ~b,~b); */\* Объясните, как*

*работает операция инверсия \*/*

printf("**\n**");

*/\* 0x%x=%d - В строке форматов 0x является частью текста,*

*%x - спецификатор формата для вывода шестнадцатеричного числа*

*= - часть текста*

*%d - спецификатор формата для вывода целого числа*

*При инверсии значение каждого бита числа в его двоичном представлении*

*изменяется на противоположное \*/*

printf("Операция 'сдвиг <<2' в числе a : 0x%x=%d**\n**", a<<2,a<<2); */\* Пропишите в*

*двоичном коде операцию сдвига на конкретном примере\*/*

printf("Операция 'сдвиг >>2' в числе a : 0x%x=%d**\n**", a>>2,a>>2); */\* Пропишите в*

*двоичном коде операцию сдвига на конкретном примере\*/*

printf("**\n**");

*/\* 0000000000101110<<2=0000000010111000*

*0000000000101110>>2=0000000000001011 \*/*

printf("Операция 'битовое И' a&b : 0x%x=%d**\n**", a&b,a&b);

*/\* Объясните, как работает операция битовое И*

*Последовательно применяется логическое умножение для каждой пары*

*битов операндов. В случае если оба бита равны 1, результат равен 1.*

*в остальных случаях результат 0\*/*

printf("Операция 'битовое ИЛИ' a|b : 0x%x=%d**\n**", a|b,a|b);

*/\* Объясните, как работает операция битовое ИЛИ*

*Последовательно применяется логическое сложение для каждой пары*

*битов операндов. Результат равен 1 если хотя бы один из битов равен 1.*

*В остальных случаях результат 0\*/*

printf("Операция 'битовое исключающее ИЛИ ': 0x%x=%d**\n**", a^b,a^b);

*/\* Объясните, как работает операция битовое исключающее ИЛИ*

*Последовательно сравниваются биты операндов, используется операция*

*исключающее ИЛИ. Результат равен 1 если биты операндов различны,*

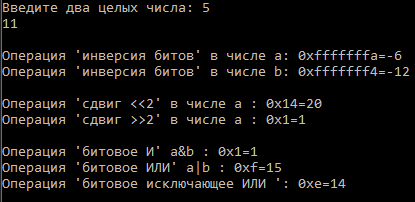
*и 0, если они имеют одинаковые значения\*/*

printf("**\n**");

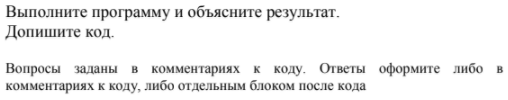
getch();

}

Результат:



Задание 2.



Код:

#include<stdio.h>

#include<conio.h>

#include<windows.h>

**void** main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

**int** i,a,b;

*/\* —————————————————— \*/*

printf("**\n**Введите два целых числа: "); scanf("%d %d",&a,&b);

printf("**\n**");

printf("Номера и значения битов старшего байта числа a:**\n**");

**for** (i=31;i>=24;i--)

*/\* Объясните используемый интервал*

*В связи с компиляцией кода в системе, где размер int составляет*

*4 байта, номера битов были изменены в соответствии с исходным*

*назначением. Границы данного интервала отображают номера перебираемых битов\*/*

printf("(%d,%d) ",i,(a>>i)&0x01); */\* Объясните запись (a>>i)&0x01 \*/*

printf("**\n\n**");

*/\* a смещается на i битов вправо и побитово умножается на 01.*

*Таким образом из двоичной записи числа поочередно слева направо*

*извлекается каждый бит. Инструкция &0x01 служит для отделения последнего*

*бита после сдвига последовательности\*/*

printf("Номера и значения битов младшего байта числа a:**\n**");

**for** (i=7;i>=0;i--)

*/\* Объясните используемый интервал \*/*

*/\* Интервал, описывающий позиции битов младшего байта числа \*/*

printf("(%d,%d) ",i,(a>>i)&0x01);

printf("**\n\n**");

printf("Биты числа b в прямом порядке : ");

**for** (i=15;i>=0;i--)

printf("%d",(b>>i)&0x01);

printf("**\n**");

*/\* ————————————– \*/*

printf("Биты числа b в обратном порядке: "); */\* Допишите код, ссылаясь на*

*пример: «Биты числа b в прямом порядке»*

*for (;;)*

*printf("",);*

*printf("\n\n");\*/*

**for** (i=0;i<=15;i++)

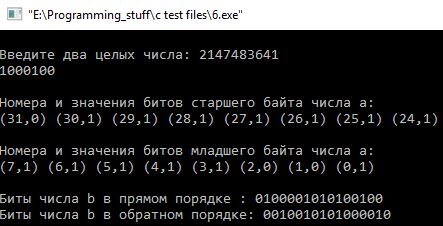
printf("%d",(b>>i)&0x01);

printf("**\n**");

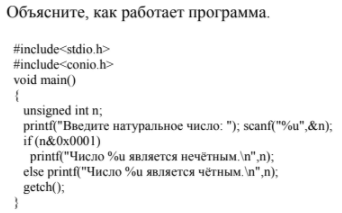
getch();

}

Результат:



Задание 3.



Код:

#include<stdio.h>

#include<conio.h>

#include<windows.h>

**void** main()

{

SetConsoleCP(CP\_UTF8);

SetConsoleOutputCP(CP\_UTF8);

**unsigned** **int** n; *// Объявляется беззнаковая целочисленная переменная n*

printf("Введите натуральное число: "); scanf("%u",&n);

*/\* Введённое число считывается в переменную n с соответствующим*

*модификатором ввода \*/*

**if** (n&0x0001) */\* Для того, чтобы узнать, является ли число чётным,*

*достаточно знать значение последнего бита. В данной строке как раз и*

*проверяется значение последнего бита путём побитового умножения. Если в*

*нулевой бит равен 1 - число нечётное, при 0, соответственно, наоборот\*/*

printf("Число %u является нечѐтным.**\n**",n);

**else** printf("Число %u является чѐтным.**\n**",n);

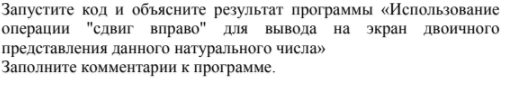
getch();

}

Результат:



Задание 4.



Код:

#include<conio.h>

#include<locale.h>

**void** main()

{

setlocale(LC\_ALL,"Russian");

**long** **int** a; */\* Исходное натуральное число:*

*если a<255, то для хранения числа требуется 7 и менее бит памяти;*

*если a<65535, то для хранения числа требуется менее 15 и менее памяти;*

*если a<2147483647, то для хранения числа требуется менее 31 и менее бит памяти \*/*

**int** i, n; *// Параметр цикла*

printf("Введите натуральное число : "); scanf("%lu",&a);

n=(a>255)?(a>65535)?31:15:7; *// объясните это выражение*

*/\* Два тернарных оператора, один вложен в другой. При ложности условия*

*a>255 n присваивается значение 7, при выполнении условия происходит проверка условия a>65535, в случае истинности которого n присваивается значение 31, иначе – 15.*

*\*/*

printf("Его двоичное представление: ");

**for** (i=n;i>=0;i--)

{

printf("%d",(a>>i)&0x1); *// объясните выражение (a>>i)&0x1*

*/\* Сдвиг a на i бит, т.е. последовательно на каждой итерации цикла*

*происходит смещение i-го бита в конец числа и побитовое умножение на 0x01*

*для вычленения бита \*/*

**if** (i%8==0) *//объясните, для чего служит эта операция*

*/\* Проверка на кратность 8, для отделения байтов \*/*

printf(" ");

}

printf("**\n**");

getch();

}

Результат:

